Basic Commands of UNIX Operating System

Sonali Grover,
Department of Information technology,
Dronacharya College of Engineering, Gurgaon, India

***Abstract*- UNIX is a computer operating System which is capable of handling activities from multiple users at the same time. In computing, commands are a directive to a computer program acting as an interpreter of some kind, in order to perform a specific task. This paper gives a brief introduction of UNIX operating system and covers some of the basics commands of UNIX operating system.**

***Index terms*- UNIX, commands, operating system.**

I. INTRODUCTION

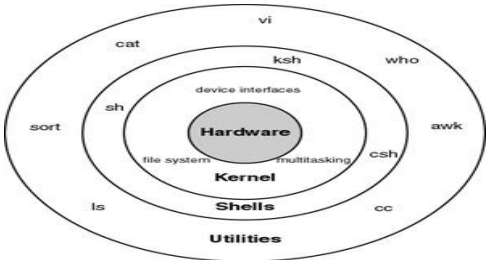
The UNIX operating system is a set of programs that act as a link between the computer and the user.

The computer program that allocates the system resources and coordinates all the details of the computer's internals is called the operating system or kernel.

Users communicate with the kernel through a program known as the shell. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

- Unix was originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna.
- There are various Unix variants available in the market. Solaris Unix, AIX, HP Unix and BSD are few examples. Linux is also a flavour of Unix which is freely available.
- Several people can use a UNIX computer at the same time; hence UNIX is called a multiuser system.
- A user can also run multiple programs at the same time; hence UNIX is called multitasking.

II. UNIX ARCHITECTURE



The main concept that unites all versions of UNIX is the following four basics:

- **Kernel:** The kernel is the heart of the operating system. It interacts with hardware and most of the tasks like memory management, tash scheduling and file management.
- **Shell:** The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are most famous shells which are available with most of the Unix variants.
- **Commands and Utilities:** There are various command and utilities which you would use in your day to day activities. **cp, mv, cat** and **grep** etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various optional options.
- **Files and Directories:** All data in UNIX is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system.

III. BASIC COMMANDS

➢ cal

This command will print a calendar for a specified month and/or year.

- To show this month's calendar, enter:

```
cal
```

- To show a twelve-month calendar for 2012, enter:

```
cal 2012
```

- To show a calendar for just the month of June 1970, enter:

```
cal 6 1970
```

- **cat**

This command outputs the contents of a text file. You can use it to read brief files or to concatenate files together.

- To append file1 onto the end of file2, enter:

```
cat file1 >> file2
```

- To view the contents of a file named myfile, enter:

```
cat myfile
```

- **cd**

This command changes your current directory location. By default, your Unix login session begins in your home directory.

- To switch to a subdirectory (of the current directory) named myfiles, enter:

```
cd myfiles
```

- To switch to a directory named /home/dvader/empire_docs, enter:

```
cd /home/dvader/empire_docs
```

- To move to the parent directory of the current directory, enter:

```
cd ..
```

- To move to the root directory, enter:

```
cd /
```

- To return to your home directory, enter:

```
cd
```

- **chmod**

This command changes the permission information associated with a file. Every file (including directories, which Unix treats as files) on a Unix system is stored with records indicating who has permission to read, write, or execute the file, abbreviated as r, w, and x. These permissions are broken down for three categories of user: first, the owner of the file; second, a group with which both the user and the file may be associated; and third, all other users. These categories are abbreviated as u for owner (or user), g for group, and o for other.

- To allow yourself to execute a file that you own named myfile, enter:

```
chmod u+x myfile
```

- To allow anyone who has access to the directory in which myfile is stored to read or execute myfile, enter:

```
chmod o+rx myfile
```

- **cp**

This command copies a file, preserving the original and creating an identical copy. If you already have a file with the new name, cp will overwrite and destroy the duplicate. For this reason, it's safest to always add -i after the cp command, to force the system to ask for your approval before it destroys any files.

- The general syntax for cpis:

```
cp -i oldfile newfile
```

- To copy a file named meeting1 in the directory /home/dvader/notes to your current directory, enter:

```
cp -i /home/dvader/notes/meeting1 .
```

The . (period) indicates the current directory as destination, and the -i ensures that if there is another file named meeting1 in the current directory, you will not overwrite it by accident.

- To copy a file named oldfile in the current directory to the new name newfile in themystuff subdirectory of your home directory, enter:

```
cp -i oldfile ~/mystuff/newfile
```

The ~ character (tilde) is interpreted as the path of your home directory.

- **date**

The date command displays the current day, date, time, and year.

```
date
```

- **df**

This command reports file system disk usage (i.e., the amount of space taken up on mounted file systems). For each mounted file system, df reports the file system device, the number of blocks used, the number of blocks available, and the directory where the file system is mounted.

- To find out how much disk space is used on each file system, enter the following command:

```
df
```

- If the df command is not configured to show blocks in kilobytes by default, you can issue the following command:

```
df -k
```

➢ **du**

This command reports disk usage (i.e., the amount of space taken up by a group of files). The du command descends all subdirectories from the directory in which you enter the command, reporting the size of their contents, and finally reporting a total size for all the files it finds.

- To find out how much disk space your files take up, switch to your home directory with the cd command, and enter:

```
du
```

The numbers reported are the sizes of the files; on different systems, these sizes will be in units of either 512 byte blocks or kilobytes. To learn which the case is, use the man command, described below. On most systems, du -k will give sizes in kilobytes.

➢ **find**

The find command lists all of the files within a directory and its subdirectories that match a set of conditions. This command is most commonly used to find all of the files that have a certain name.

- To find all of the files named myfile.txt in your current directory and all of its subdirectories, enter:

```
find . -name myfile.txt -print
```

- To look in your current directory and its subdirectories for all of the files that end in the extension .txt, enter:

```
find . -name "*.txt" -print
```

- In these examples, the . (period) represents your current directory. It can be replaced by the full pathname of another directory to search. For instance, to search for files named myfile.txt in the directory /home/user/myusername and its subdirectories, enter:

```
find /home/user/myusername/ -name myfile.txt -print
```

On some systems, omitting the final / (slash) after the directory name can cause find to fail to return any results.

- As a shortcut for searching in your home directory, enter:

```
find "$HOME/" -name myfile.txt -print
```

➢ **Jobs**

- This command reports any programs that you suspended and still have running or waiting in the background (if you had pressed Ctrl-z to suspend an editing session, for example). For a list of suspended jobs, enter:

```
jobs
```

- Each job will be listed with a number; to resume a job, enter % (percent sign) followed by the number of the job. To restart job number two, for example, enter:

```
%2
```

This command is only available in the csh, bash, tcsh, and ksh shells.

➢ **kill**

Use this command as a last resort to destroy any jobs or programs that you suspended and are unable to restart. Use the jobs command to see a list of suspended jobs. To kill suspended job number three, for example, enter:

```
kill %3
```

- Now check the jobs command again. If the job has not been cancelled, harsher measures may be necessary. Enter:

```
kill -9 %3
```

➢ **less and more**

Both less and more display the contents of a file one screen at a time, waiting for you to press the Spacebar between screens. This lets you read text without it scrolling quickly off your screen. The less utility is generally more flexible and

powerful than more, but more is available on all Unix systems while less may not be.

- To read the contents of a file named textfile in the current directory, enter:

```
less textfile
```

- The less utility is often used for reading the output of other commands. For example, to read the output of the ls command one screen at a time, enter:

```
ls -la | less
```

In both examples, you could substitute more for less with similar results. To exit either less or more, press q. To exit less after viewing the file, press q.

Note: Do not use less or more with executables (binary files), such as output files produced by compilers. Doing so will display garbage and may lock up your terminal

- **lpr and lp**

These commands print a file on a printer connected to the computer network. Thelpr command is used on BSD systems, and the lp command is used in System V. Both commands may be used on the UITS systems.

To print a file named myfile on a printer named lp1 with lpr, enter:

```
lpr -Plp1 myfile
```

To print the same file to the same printer with lp, enter:

```
lp -dlp1 myfile
```

- **rm**

This command will remove (destroy) a file. You should enter this command with the -i option, so that you'll be asked to confirm each file deletion.

To remove a file named junk, enter:

```
rm -i junk
```

- **mkdir**

This command will make a new subdirectory.

- To create a subdirectory named mystuff in the current directory, enter:

```
mkdir mystuff
```

- To create a subdirectory named morestuff in the existing directory named /tmp, enter:

```
mkdir /tmp/morestuff
```

- **Mv**

This command will move a file. You can use mv not only to change the directory location of a file, but also to rename files. Unlike the cp command, mv will not preserve the original file.

Note: As with the cp command, you should always use -i to make sure you do not overwrite an existing file.

To rename a file named oldname in the current directory to the new name newname, enter:

```
mv -i oldname newname
```

- **ps**

The ps command displays information about programs (i.e., processes) that are currently running. Entered without arguments, it lists basic information about interactive processes you own. However, it also has many options for determining what processes to display, as well as the amount of information about each. Like lpand lpr, the options available differ between BSD and System V implementations. For example, to view detailed information about all running processes, in a BSD system, you would use ps with the following arguments:

```
ps -alxww
```

To display similar information in System V, use the arguments:

```
ps -elf
```

- **rmdir**

This command will remove a subdirectory. To remove a subdirectory named oldfile, enter:

```
rmdir oldfile
```

- **vi**

This command starts the vi text editor. To edit a file named myfile in the current directory, enter:

```
vi myfile
```

- **set**

This command displays or changes various settsings and options associated with your Unix session.
To see the status of all settings, enter the command without options:

```
set
```

If the output scrolls off your screen, combine set with less:

```
set | less
```

The syntax used for changing settings is different for the various kinds of Unix shells; see the man entries for set and the references listed at the end of this document for more information.

- **w and who**

The w and who commands are similar programs that list all users logged into the computer. If you use w, you also get a list of what they are doing. If you use who, you also get the IP numbers or computer names of the terminals they are using.

REFERENCES

[1] https://kb.iu.edu/d/afsk
[2] http://www.tutorialspoint.com
[3] http://www.tjhsst.edu/~dhyatt/superap/unixcmd.html
[4] http://en.wikipedia.org/wiki/Unix
[5] http://mally.stanford.edu/~sr/computing/basic-unix.html